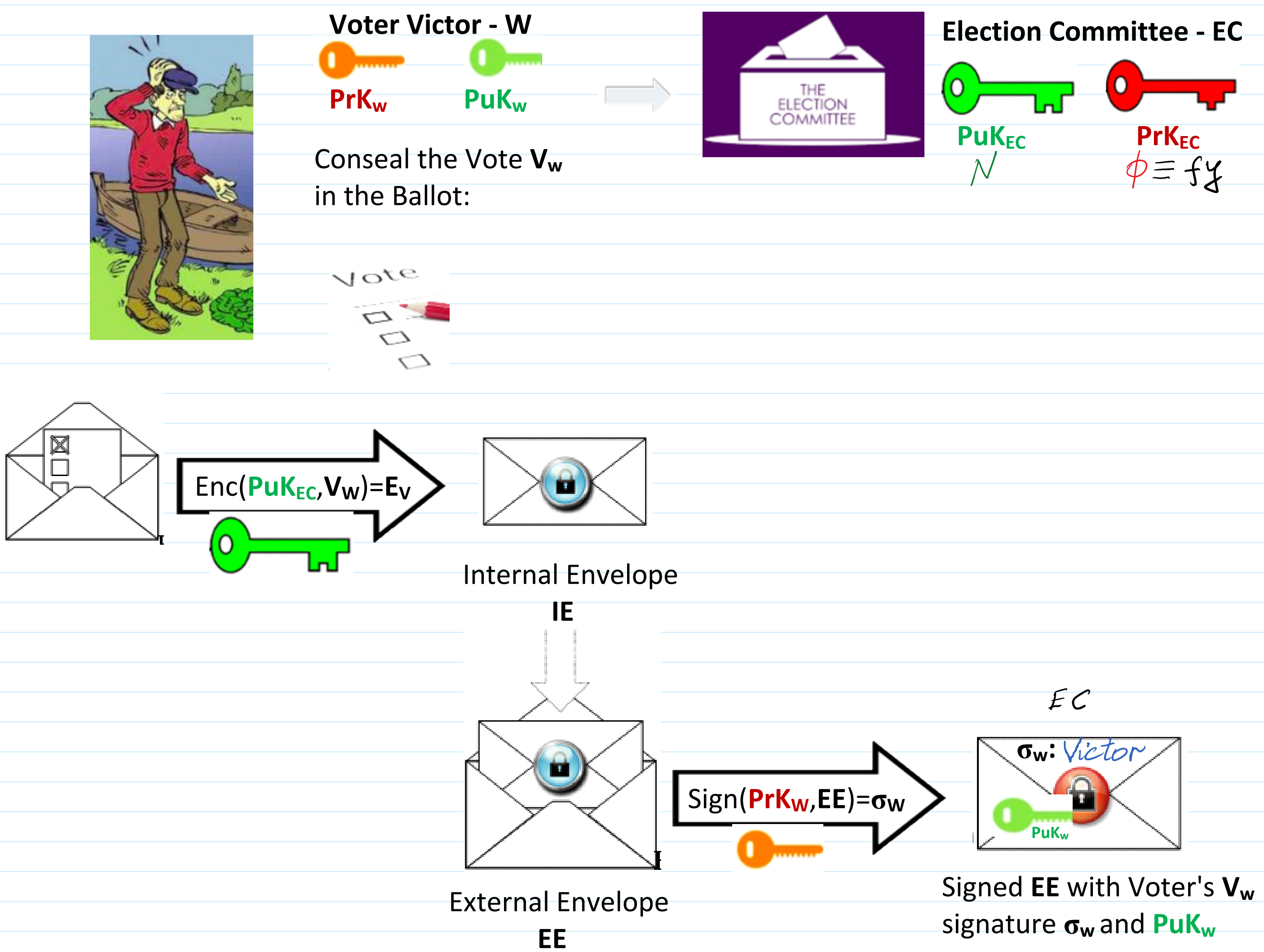
eVoting System must guarantee:
- Conseal the Vote
- Conseal the Ballots

Table of eVoting in Google drive
https://docs.google.com/spreadsheets/d/1joAuG6oh1arcQqc7zPTPbLBH48wG8Fn6/edit?usp=sharing&ouid=111502255533491874828&rtpof=true&sd=true



After eVoting it is the time to calculate the results.

1. **EC** verifies Voter's V_w PuK_w and if PuK_W is registered in **EC** database then goes to step 2.
2. **EC** verifies PuK_w certificate and if it is valid then goes to step 3.
3. **EC** verifies signature σ_w on **EE** and if it is valid then extracts **IE** and prceeds with ballots computation.

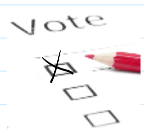
```
>> p=109;
>> q=127;
>> N=p*q                                    % PuKEC=NEC=13843
N = 13843
                                            % |N|=14 bits
```

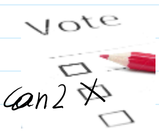
```
>> N_2=int64(N*N)
N_2 = 191628649
>> dec2bin(N_2)
ans = 1011 0110 1100 0000 0101 0110 1001

>> fy=(p-1)*(q-1)
fy = 13608                                        % PrK_EC=fy
```

Ballots computation

1. Collects all encrypted votes: $(E_W, E_2, E_3, \dots, E_M)$.
   Number of Voters is M.
2. Multiplies all encrypted votes
   $$E = E_W \cdot E_2 \cdot E_3 \cdot \dots \cdot E_M \mod N^2$$
3. Decrypts E
   $$Dec(PrK_{EC}, E) = V.$$

If there are 2 candidates: Can1 := 0 ; Can2 := 1

0 ≡ can1 

1 ≡ can2 

When using Paillier homomorphic encryption

$$Dec(PrK_{EC}, E) = V = V_W + V_2 + V_3 + \dots + V_M$$

Let V_{k1} is a number of votes dedicated to Can 1.
Let V_{k2} is a number of votes dedicated to Can 2: $\Rightarrow V = V_{k2}$.
Then the number of votes for Can1: $M - V$

We used homomorphic encryption property:

$$Enc(PuK_{EC}, V_W + V_2 + V_3 + \dots + V_M) = E_W \cdot E_2 \cdot E_3 \cdot \dots \cdot E_M \mod N^2 = E$$

where $E_W = Enc(PuK_{EC}, V_W)$, $E_2 = Enc(PuK_{EC}, V_2)$, ----

$$Dec(PrK_{EC}, E) = Dec(PrK_{EC}, E_W \cdot E_2 \cdot E_3 \cdot \dots \cdot E_M) = V_W + V_2 + V_3 + \dots + V_M = V$$

Let: **K** - be a number of Candidates (Can); for example.

M - be a number of Voters (V); **M**=15.
For every candidate **Can1**, **Can2**, …, **CanK** the **Vote** is encoded by certain integer number.
Since all **Votes** are encrypted by every **Voter** using Paillier homomorphic encryption scheme, therefore the maximal sum of **Votes** must not increase PuK_{EC} value **N**.
It is due to the property of Paillier encryption stating that encrypted message $m \in Z_N=\{0, 1, 2, 3, \ldots, N-1\}$.
Then due to homomorphic property of Paillier encryption when all encrypted **Votes** are multiplied the obtained result **E** (computed mod N^2) can be correctly decrypted and indicate the sum of all **Votes**.
Then encoding to **Votes** for every candidate they must be chosen in such a way that they can be distinguished from the sum of Votes of other candidate.

Let us consider three candidates **Can1**, **Can2**, **Can3** for our generated $PuK_{EC}=N$=13843, |N|=14 bits.
For **Votes** separation of 3 **Candidates** we assign the total sum of **Votes** is represented by 4 bits for every candidate.
This sum can be achieved by optimal encoding of **Votes** consisting of the following cases.

1. The **Vote** for **Can1** is encoded by number 2^8=**256**. Then If all 15 **Voters** vote for **Can1** the total sum of **votes** will be 15∗256=3840. Notice that 3840+256=4096=2^{12}.
2. The **Vote** for **Can2** is encoded by number 2^4=**16**. If all 15 **Voters** vote for **Can2** the total sum will be 15∗16=240. Notice that 240+16=256=2^8.
3. The **Vote** for **Can3** is encoded by number **1**. If all 15 **Voters** vote for **Can1** the total sum will be 15 = 1111_b.

Then the maximal sum of votes is obtained in the case 1 and is equal to 3840 < 14351=**N**.
In tables below the maximal sum of Votes for **Can1**, **Can2**, **Can3** encoded in binary with 4 bit length is presented.
Then the maximal sum of **Voters** votin for the candidate **Can1** can not exceed number 15=2^4-1=1111_b.

0	0	0	0	0	0	0	0	0	0	0	1
Can1				Can2				Can3			

For **Can3**: 0000 0000 0001_b=1

0	0	0	0	0	0	0	1	0	0	0	0
Can1				Can2				Can3			

For **Can2**: 0000 0001 0000_b=2^4 =16

0	0	0	1	0	0	0	0	0	0	0	0
Can1				Can2				Can3			

For **Can1**: 0001 0000 0000_b=2^8 =256

Sum of total votes for every candidate:

0	0	0	0	0	0	0	0	1	1	1	1
Can1				Can2				Can3			

For **Can3**: 0000 0000 1111_b=15

0	0	0	0	1	1	1	1	0	0	0	0
Can1				Can2				Can3			

For **Can2**: 0000 1111 0000_b=240

1	1	1	1	0	0	0	0	0	0	0	0
Can1				Can2				Can3			

For **Can1**: 1111 0000 0000_b=3840

The Globe wide Voting

Let us imagine that election is performed in the half of the Globe with number of **Voters M** is about 4 billions.
Let **M** < 2^{32} = 4 294 967 296.
Let the number of **Candidates** to be elected is about 1000.
Let **K** < 2^{10} = 1 024.
Then the number of bits for election data representation for every of 1024 = 2^{10} **Candidates** is
$2^{10} * 2^{32} = 2^{42}$ = 4 398 046 511 104 and is about 4 trillions.
Then the maximal sum of **Votes** is **K**∗**M** and is represented by 2^{42} = 4 398 046 511 104 bits number and is corresponding to the decimal number (2)^(2^{42}) - 1 = $2^{4\ 398\ 046\ 511\ 104}$ - 1.
Since the sum of **Votes** must be less than **PuK**$_{EC}$=**N**, then **N** must be close to the number $2^{4\ 398\ 046\ 511\ 104}$ - 1.
Then |**N**|= 4 398 046 511 104 bits.
Since **N**=**p**∗**q**, where **p**, **q** are primes, then |**p**|=|**q**|= 2 199 023 255 552 bits.
The problem is to generate such a big prime numbers.
If we encode decimal numbers in ASCII code then 1 decimal digit is encoded by 8 bits.
Then **p**, **q** numbers in decimal representation will have 2 199 023 255 552 / 8 = 274 877 906 944 decimal digits.
It is more than 274 billions.

Problem solution.

The solution is to divide election into different **Voting Areas** so reducing number of **Voters M**.
Then encryption scheme becomes more practical and more efficiently realizable.
Let we are able to generate considerable large prime numbers **p**, **q** having 2^{15} = 32 768 bits,
i.e. |**p**|=|**q**|= 2^{15} = 32 768 bits and hence are bounded by 2^{32768} - 1 such a huge decimal number.
Notice that in traditional cryptography for prime numbers it is enough to have 4096 bit length.
Then **N**=**p**∗**q** will have 32 768 + 32 768 = 65 536 = 2^{16} bit length and hence is bounded by the following $2^{65\ 536}$ - 1 huge decimal number.
Then the arithmetic operations are performed with such a huge numbers and even with numbers up to **N^2** since operations **mod N^2** are used. Therefore the special software is needed.
Let **Voting Areas** are divided in such a way that they can serve about 16 millions **Voters**.
Assume that number of **Voters M** < 16 777 215 = 2^{24} - 1. Then |**M**| = 24 bits.
Then for every candidate we must dedicate 24 bits in the total string of bits of number **PuK**$_{EC}$=**N** where |**N**|=2^{16} = 65 536 bit length.
Then number of **Candidates K** in **Voting Area** is the following:

$$K = |N| \;/\; |M| = 2^{16} / 24 = 2731.$$

The distribution of **Candidates** and the number of bits them assigned is presented in table.

Total length of **N** is 65 536 bits

24 bits	24 bits	24 bits		24 bits
Can1	**Can2**	**Can3**	⟵——⟶	**Can2731**

There are 2 problems must be solved:
1. To generate 2 large prime numbers **p**, **q** having 2^{15} = 32 768 bit length ~ 10^{10000} : it is feasible.
2. To perform a computations with large numbers using special software having 2^{32} = 4 294 967 296 bits when operations are performed **mod N^2** .

Practical exercises

The total maximal number of sum of votes with 15 voters is 3840 < **N** = 13843.

```
>>  p=109;
>> pb=dec2bin(p)
pb = 1101101
>> q=127;
>> qb=dec2bin(q)
qb = 1111111
>> N=p*q
N = 13843
>> Nb=dec2bin(N)
Nb = 11011000010011
>> N_2=int64(N*N)
N_2 = 191628649
>> dec2bin(N_2)
ans = 1011 0110 1100 0000 0101 0110 1001
```

% **PuK_{EC}**=**N**=13843
% |N|=14 bits

```
>> fy=(p-1)*(q-1)
fy = 13608
```

% **PrK=fy**

- **Enc**: on input a public key N and a message $m \in \mathbb{Z}_N$, choose a random $r \leftarrow \mathbb{Z}_N^*$ and output the ciphertext

$$c := [\underbrace{(1+N)^m}_{e_1 \bmod N^2} \cdot \underbrace{r^N}_{e_2 \bmod N^2} \bmod N^2].$$

$$e_w = c = e = e_1 \cdot e_2 \bmod N^2$$

$$\mathbb{Z}_N^* = \{ z \mid \gcd(z, N) = 1 \}$$

$$z < N$$

```
>> vw=16
vw = 16
>> rw=randi(N-1)
rw = 5029
>> gcd(rw,N)
ans = 1
>> ew1=mod_exp((1+N),vw,N_2)
ew1 = 221489
>> ew2=mod_exp(rw,N,N_2)
ew2 = 115257872
>> ew=mod(ew1*ew2,N_2)
ew = 157077575
```

```
>> E=mod(ew*e2,N_2)
E = 108508702
```

```
>> w2=256
v2 = 256
>> r2=randi(N-1)
r2 = 12539
>> gcd(r2,N)
ans = 1
>> e21=mod_exp((1+N),v2,N_2)
e21 = 3543809
>>  e22=mod_exp(r2,N,N_2)
e22 = 57431777
>> e2=mod(e21*e22,N_2)
e2 = 184773534
```

$$c^{\phi} \bmod N^2 = d_1$$

$$\phi^{-1} \bmod N = d_3$$

$$m := \left[\underbrace{\frac{[c^{\phi(N)} \bmod N^2] - 1}{N}}_{\frac{d_1 - 1}{N} \bmod N = d_2} \cdot \underbrace{\phi(N)^{-1}}_{} \bmod N \right].$$

$$m = d_2 \cdot d_3 \bmod N$$

```
>> d1=mod_exp(E,fy,N_2)
```

```
>> d1=mod_exp(E,fy,N_2)
d1 = 73298686
>> d2=mod((d1-1)/N,N)
ans = 8462
>> d2=mod((d1-1)/N,N)
d2 = 5295
>> fy_m1=mulinv(fy,N)
fy_m1 = 1885
>> d3=fy_m1
d3 = 1885
>> m=mod(d2*d3,N)
m = 272
>> V=m
V = 272
```

Can1 := 256

```
>> NVCan1=floor(272/256)
NVCan1 = 1
>> 272/256
ans = 1.0625
```

Can2 := 16

```
>> SumVCan2=V-1*256
SumVCan2=16
NVCan2 = floor(SumVCan2/16)
NVCan2=1
```

Can3 := 1

Election examole by students

Jokūbas Žitkevičius 15:44
E1=68076817 Can3=1

Melita 15:44
E2=158874063 Can2=16

Egidijus Sinkevičius
E3=157077575 Can2=16

Silvija Petkeviciute 15:45
E4=182226291 Can1=256

The multiplication result of Encrypted votes:
$E=E1*E2*E3*E4 \bmod N^2 = 114046$

Can1=4, Can2=10, Can3=8
1192= 4*256+10*16+8*1

1, 16, 16, 256
1*256+2*16+1*1 = 289

```
>> v1 = pai_dec(fy, N, E1)
v1 = 1
>> v2 = pai_dec(fy, N, E2)
v2 = 16
>> v3 = pai_dec(fy, N, E3)
v3 = 16
>> v4 = pai_dec(fy, N, E4)
v4 = 256
```

```
>> E12=mod(E1*E2,N_2)
E12 = 178689144
>> E34=mod(E4*E3,N_2)
E34 = 33756858
>> E1234=mod(E12*E34,N_2)
E1234 = 159883416
```

```
>>  v1234 = pai_dec(fy, N, E1234)
v1234 = 289
```

P.Vardas	No	ri	ci	c	V_by_M: cMi	c*cMi	Dec(c*cMi)	Tot_S_of_V
Au. Juozas	1	16339	149318501	216987098	92831661	152067656	896	896
Be. Antanas	2	8609	32143614	216987098	123083220	203234256	896	896
	3							
	4							
	5							
	6							
	7							
	8							

	9	
	10	
	11	
ri	12	Random number generated for Paillier encryption
ci	12	Your vote encrypted by Paillier encryption
c	14	The product of all encrypted votes in your polling station. **Provided by lecturer**
V_by_M: cMi	15	Encrypted Vote received by Mail: cMi. **Provided by lecturer**
c*cMi		Multiplied encrypted votes in polling station multiplied by cMi
Dec(c*cMi)		Decryption all multiplied votes
Tot_S_of_V		Total sum of votes

No	N_of_V_Can1	N_of_V_Can2	N_of_V_Can3	Tot_N_of_V	Dec(cMi)	**Acc/Dec** cMi	Can1	Can2	Can3
1	5	8	0	13	512, 2 balsai uz pirma	Dec	3	8	0
2	6	8	0	14	256, 256, 256	Dec	3	8	0
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									
12									
14									
15									

N_of_V_Can1		Number of votes for Can1
N_of_V_Can2		Number of votes for Can2
N_of_V_Can3		Number of votes for Can3
Tot_N_of_V		Total number of votes
Dec(c*Mi)		Decrypted vote cMi received by Mail
Acc/Dec cMi		Accept or Decline vote received by Mail. Input: **Acc** or **Dec**
Can1		Number of votes for Can1
Can2		Number of votes for Can2
Can3		Number of votes for Can3

$M = 15$ Voters: $V = 2212 = N_1(\overset{256}{Can1}) + N_2(\overset{16}{Can2}) + N_3(\overset{1}{Can3})$

0	0	0	0	0	0	0	0	0	0	0	1
Can1				**Can2**				**Can3**			

For **Can3**: $0000\ 0000\ 0001_b=1$

0	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**
Can1				**Can2**				**Can3**			

For **Can2**: 0000 0001 $0000_b = 2^4 = 16$

0	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Can1				**Can2**				**Can3**			

For **Can1**: 0001 0000 $0000_b = 2^8 = 256$

Till this place

1)

2